1 Meow game app

1.1 Interactive sound

Type in the following code, save it and test it:

```
function love.load()
  sound = love.audio.newSource( "meow1.ogg" )
end

function love.mousepressed()
  sound:play()
end
```

The code in `love.load()` loads a sound file, `love.mousepressed()` plays it when a mouse button is pressed or the touchscreen is touched.

1.2 Interactive image

Add the loading of two images to `love.load()`:

```
  bild_auf = love.graphics.newImage( "open.png" )
  bild_zu = love.graphics.newImage( "closed.png" )
```

Insert the following two functions to your code:

```
function love.update()
  bild_aktuell = bild_zu
  if sound:isPlaying() then bild_aktuell = bild_auf end
end

function love.draw()
  love.graphics.draw( bild_aktuell, 0, 0 )
end
```

`love.update()` calculates, which of the images is the current one. `love.draw()` draws it. Both functions work 60 times per second. The image doesn't quite fit but we will take care of that later.

1.3 Random meow sounds

Add the following random generator initiator and list (or table) of sounds to `love.load()`:

```
  soundliste = {
    love.audio.newSource( "meow1.ogg" ),
    love.audio.newSource( "meow2.ogg" ),
    love.audio.newSource( "meow3.ogg" ),
    love.audio.newSource( "meow4.ogg" ),
    love.audio.newSource( "meow5.ogg" ),
  }
  math.randomseed( os.time() )
```

Replace the content of `love.update()` with code, which uses the sound list:

```
  bild_aktuell = bild_zu
  for i,u in pairs(soundliste) do
    if u:isPlaying() then bild_aktuell = bild_auf end
  end
```

Replace the content of `love.mousepressed()` with code which plays random sounds:

```
  wahl = math.random(1,5)
  soundliste[wahl]:stop()
  soundliste[wahl]:play()
```

1.4 Adapt to different screens

Add calculations of the relations between image and window size to `love.load()`:

```
  fx = love.graphics.getWidth() / 1024
  fy = love.graphics.getHeight() / 600
```

Add scaling parameters to the `love.graphics.draw()` function call in `love.draw()`:

```
  love.graphics.draw(bild_aktuell, 0, 0, 0, fx, fy)
```

The image fits to the screen size this way, since mobile phones/tablets only have one resolution. This is not optimal but a simple solution for the start.

1.5 Android port

If you would like to put own graphics (you can draw on the computer or on paper) and sounds into your meow game app, tell the workshop coaches. You can also change the app icon.

We recommend to code the "back" button to close the Android app:

```
function love.keypressed( key )
  if key == "escape" then love.event.quit() end
end
```

To make the app playable on Android, a zip archive of the game has to be made, it must be renamed to `game.love` and put into the project directory. Then use the `make-apk` script. The resulting `game.apk` must then be put on the mobile phone/tablet and installed there. Let workshop coaches help you.

2 Cat and mouse game app

2.1 Image and sound

Type in the following code (without `-- comments`), save it and test it:

```
function love.load()
  math.randomseed( os.time() )     -- For random numbers
  love.window.setMode( 1280, 720) -- Changes screen size
  grasBild  = love.graphics.newImage( "gras.png" )
  katzeBild = love.graphics.newImage( "katze.png" )
  mausBild  = love.graphics.newImage( "maus.png" )
  katzeX = 400 -- Position of the cat
  katzeY = 300
  mausX = 300  -- Position of the mouse
  mausY = 150
  musik = love.audio.newSource( "musik.ogg" )
  musik:setLooping( true )
  musik:play()
end

function love.draw()
  love.graphics.draw( grasBild, 0, 0 )
  love.graphics.draw( katzeBild, katzeX, katzeY )
  love.graphics.draw( mausBild, mausX, mausY )
end
```

The code in `love.load()` changes the screen resolution, loads the images and music, sets position variables and plays the msuic. `love.draw()` draws the images, 60 times per second. They don't quite fit but we will take care of that later.

2.2 Automatic and interactive movement

Add mouse click position variables and sounds to `love.load()`:

```
  klickX = 400
  klickY = 300
  quietsch = love.audio.newSource( "quietsch.ogg" )
  miau     = love.audio.newSource( "miau.ogg" )
```

Add the following three functions to your code:

```
function distanz( x1, y1, x2, y2 )
  a = x1 - x2
  b = y1 - y2
  return( math.sqrt( a^2 + b^2 ) )
end

function love.update()
  mausX = mausX + 7
  if mausX > 800 then
    mausX = -48
    mausY = math.random( 20, 400 )
  end
  if distanz( katzeX, katzeY, mausX, mausY ) < 40 then
    quietsch:play()
    mausX = 999
  end
  if distanz( katzeX, katzeY, klickX, klickY ) > 8 then
    diffX = klickX - katzeX
    diffY = klickY - katzeY
    norm = math.sqrt( diffX^2 + diffY^2 )
    einhX = diffX / norm
    einhY = diffY / norm
    katzeX = katzeX + einhX * 5
    katzeY = katzeY + einhY * 5
  end
end

function love.mousepressed( x, y )
  klickX = x
  klickY = y
  miau:play()
end
```

The `distanz()` function calculates the distance between two dots thanks to the Pythagoras' theorem or the formula $c = \sqrt{a^2 + b^2}$.

`love.update()` 1. Moves the mouse, 2. Puts the mouse back, after it crosses the right border or 3. when cat and mouse touch, 4. moves the cat

The code in `love.mousepressed()` changes the `klickX` and `klickY` variables each time a mouse button is pressed or the touchscreen is touched.

2.3 Screen size

Add calculations of the relations between image and window size to `love.load()`:

```
fx = love.graphics.getWidth() / 800
fy = love.graphics.getHeight() / 450
```

Add scaling parameters to the `love.graphics.draw()` function call in `love.draw()`:

```
love.graphics.draw( grasBild, 0, 0, 0, fx, fy )
love.graphics.draw( katzeBild, katzeX * fx, katzeY * fy, 0, fx, fy )
love.graphics.draw( mausBild, mausX * fx, mausY * fy, 0, fx, fy )
```

Ersetze die Variablenzuweisungen in `love.mousepressed()`, um vom Bildschirm aufs Spielefeld zu projezieren:

```
klickX = x/fx
klickY = y/fy
```

2.4 Score and time

Add image sizes, font configuration, time and score to `love.load()`:

```
breite = love.graphics.getWidth()
hoehe = love.graphics.getHeight()
love.graphics.setNewFont(hoehe/15)
zeitStart = love.timer.getTime()
zeit = 30
punkte = 0
```

Add time calculaation to `love.update()`:

```
zeit = 30 - math.floor(love.timer.getTime() - zeitStart)
```

Add a score counter to the one `if` block in `love.update()` which reacts to cat and mouse touching:

```
  if zeit > 0 then
    punkte = punkte + 1
  end
```

Add displaying time and score to `love.draw()`:

```
text = "Zeit: " .. zeit .. ", Punkte: " .. punkte
love.graphics.printf(text, 0, 0, breite, "center")
```

You should put the content of `love.update()` into a `if zeit > 0 then ... end` block to stop the game after the time runs out. You can use a similar block in `love.draw()` to display a "Game Over!" message.

2.5 Android port

See section 1.5.

3 Matrix music DJ app

Type in the following code (without `-- comments`), save it and test it:

```
function love.load()
  la, lg = love.audio, love.graphics
  math.randomseed( os.time() ) -- For random numbers
  namen = { "lead", "drums", "drumsb", "clap" }
  instr = {{},{}}          -- Table of instruments with...
  for i = 1, 2 do          -- two rows and...
    for j = 1, #namen do -- four columns
      instr[i][j] = {}
      instr[i][j].snd = la.newSource( namen[j] .. i .. ".ogg" )
      instr[i][j].snd:setLooping( true ) -- Endless looping on
      instr[i][j].snd:setVolume( 0 )      -- Loudness to 0
      instr[i][j].snd:play()              -- Track playback starts
      instr[i][j].farbe = { 60*j, math.random(200), 200 }
    end
  end
  spalten, zeilen = #instr[1], #instr           -- 4 columns, 2 rows
  breit, hoch = lg.getWidth(), lg.getHeight()   -- Screen size
  feldb, feldh = breit / spalten, hoch / zeilen -- Touch field size
end

function love.draw()
  for i, zeile in ipairs(instr) do  -- i is the index, zeile is the value
    for j, instrument in ipairs(zeile) do
      lg.setColor(instrument.farbe) -- Instruments have own colors
      lg.rectangle( "fill", (j-1) * feldb, (i-1) * feldh, feldb, feldh )
      if instrument.snd:getVolume() == 1 then
        lg.setColor( 255, 255, 255, 95 ) -- on/off state is displayed
        lg.circle( "fill", (j-0.5) * feldb, (i-0.5) * feldh, feldb*0.4 )
      end
    end
  end
end

function love.mousepressed(x, y) -- Gets started by mouse/touchscreen
  wob = math.ceil( x / feldb )   -- Calculating column
  woh = math.ceil( y / feldh )   -- Calculating row
  if instr[woh][wob].snd:getVolume() == 1 then
    instr[woh][wob].snd:setVolume(0) -- Loudness 0%
  else
    instr[woh][wob].snd:setVolume(1) -- Loudness 100%
  end
end
```

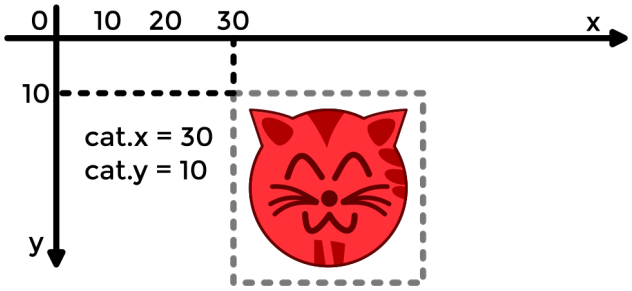
The code makes intense use of tables/lists and `for` loops as well as calculations, which might need a bit more time to be understood. Feel free to ask coaches for help.

3.1 Android port

See section 1.5.

4 Epilogue

LÖVE draws pictures and shapes using positions in a coordinate system, which originates from the top left corner towards right and down.



LÖVE is the game engine used in this workshop. More information about LÖVE programming is available at love2d.org/wiki/love .

Additional workshop material is available at espws.de/en .